R Notebook

```
library(caTools)
library(car)
```

```
## Loading required package: carData
```

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

```
library(quantmod)
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: TTR
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(MASS)
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(dplyr)
```

```
##
## ########################## Warning from 'xts' package ##########################
## #                                                                              #
## # The dplyr lag() function breaks how base R's lag() function is supposed to  #
## # work, which breaks lag(my_xts). Calls to lag(my_xts) that you type or       #
## # source() into this session won't work correctly.                            #
## #                                                                              #
## # Use stats::lag() to make sure you're not using dplyr::lag(), or you can add #
## # conflictRules('dplyr', exclude = 'lag') to your .Rprofile to stop           #
## # dplyr from breaking base R's lag() function.                                #
## #                                                                              #
## # Code in packages is not affected. It's protected by R's namespace mechanism #
## # Set `options(xts.warn_dplyr_breaks_lag = FALSE)` to suppress this warning.  #
## #                                                                              #
## ################################################################################
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:MASS':
##
##     select

## The following objects are masked from 'package:xts':
##
##     first, last

## The following object is masked from 'package:car':
##
##     recode

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
train_set <- readRDS("masq_train.Rda")
test_set <- readRDS("masq_test.Rda")
```

```
 train_X <- train_set %>%
  dplyr::select(MASQ01, MASQ02, MASQ03, MASQ04, MASQ05, MASQ06, MASQ07, MASQ08, MASQ09, MASQ11, MASQ12,

 train_Y <- train_set %>%
   dplyr::select(D_DEPDYS)

  test_X <- test_set %>%
  dplyr::select(MASQ01, MASQ02, MASQ03, MASQ04, MASQ05, MASQ06, MASQ07, MASQ08, MASQ09, MASQ11, MASQ12,

   test_Y <- test_set %>%
   dplyr::select(D_DEPDYS)

   lengths(test_X)
```

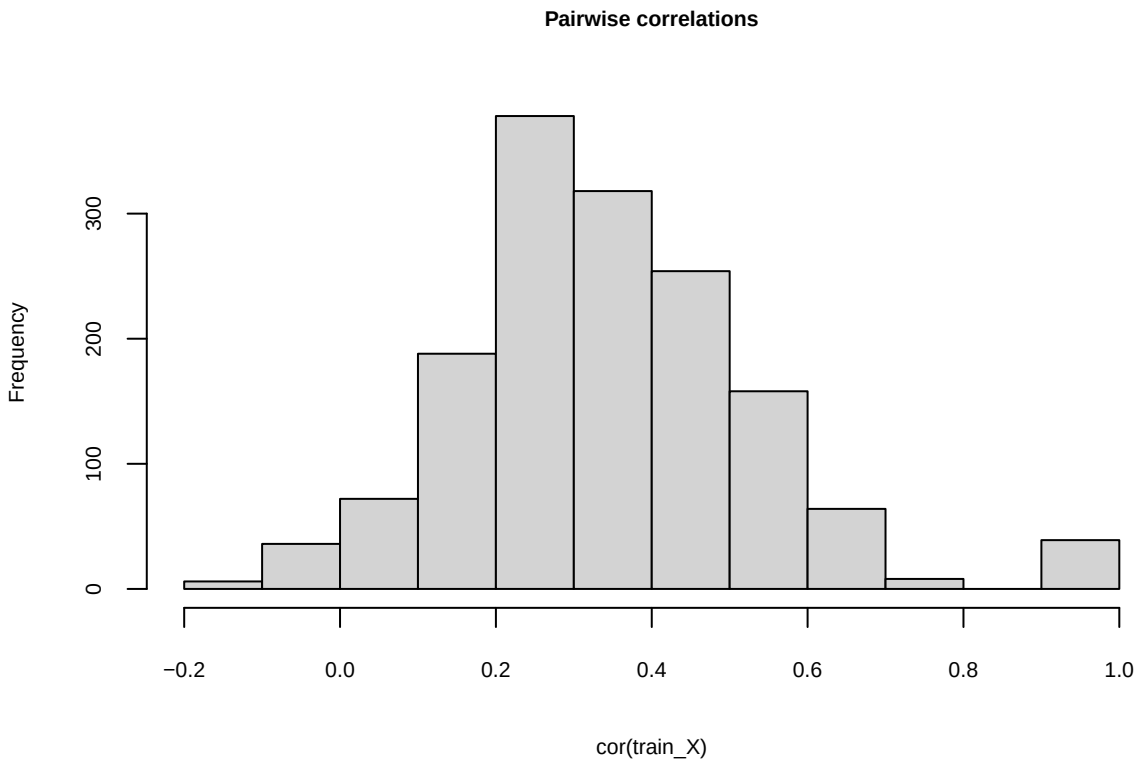
```
## MASQ01 MASQ02 MASQ03 MASQ04 MASQ05 MASQ06 MASQ07 MASQ08 MASQ09 MASQ11 MASQ12
##   1798   1798   1798   1798   1798   1798   1798   1798   1798   1798   1798
## MASQ13 MASQ14 MASQ15 MASQ16 MASQ17 MASQ18 MASQ19 MASQ20 MASQ21 MASQ22 MASQ23
##   1798   1798   1798   1798   1798   1798   1798   1798   1798   1798   1798
## MASQ24 MASQ25 MASQ26 MASQ27 MASQ28 MASQ29 MASQ30 MASQ31 MASQ32 MASQ33 MASQ34
##   1798   1798   1798   1798   1798   1798   1798   1798   1798   1798   1798
## MASQ35 MASQ36 MASQ37 MASQ38 MASQ39 MASQ40
##   1798   1798   1798   1798   1798   1798
```

```
   lengths(test_Y)
```

```
## D_DEPDYS
##     1798
```

```
#class(subset_train)

hist(cor(train_X), main = "Pairwise correlations", cex = 0.7, cex.lab = 0.7 , cex.axis = 0.7,cex.main =
```



As seen above the histogram of the correlation matrix gives a matrix that is slightly skewed to the right. This means that there are more variables that are positively correlated to each other (an increase in one variable corresponds to an increase in another variable and vice-versa). For positively correlated variables, we should use the elastic net regression.

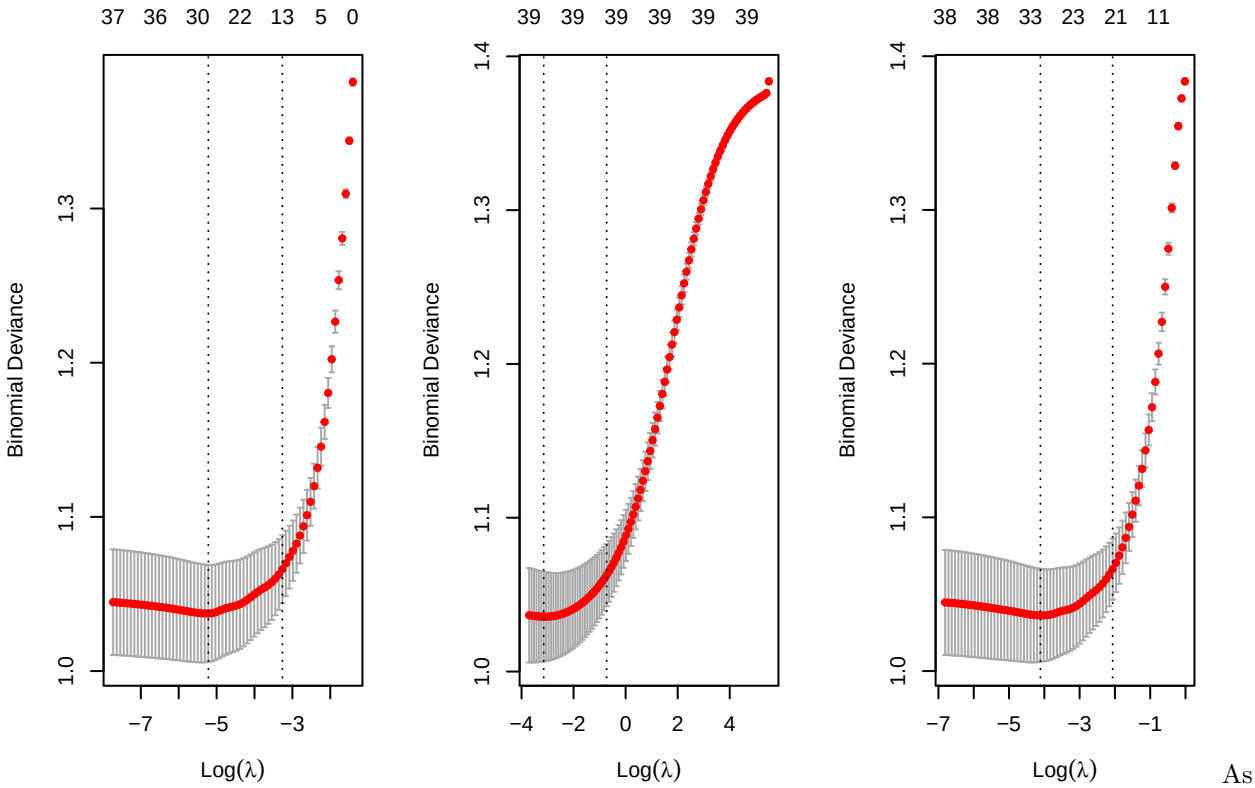
```
#set X and Y as matrices
train_X <- train_X %>% as.matrix()
train_Y <- train_Y %>% as.matrix()
test_X <- test_X %>% as.matrix()
test_Y <- test_Y %>% as.matrix()


#Lasso CV
par(mfrow = c(1, 3))
set.seed(42)
mod_l <- cv.glmnet(x=train_X, y=train_Y, family="binomial")

plot(mod_l)


#Ridge CV
set.seed(42)
mod_r <- cv.glmnet(x = train_X, y = train_Y, family="binomial", alpha = 0)
plot(mod_r)

#Elastic Net Regression
set.seed(42)
e_mod <- cv.glmnet(x=train_X, y=train_Y, family="binomial", alpha = 0.25) ## e is for elastic net e_mod
plot(e_mod)
```



As we can see the graphs for lasso and elastic net regression (left and right) have an increasing gradient shown by their convex shape. In comparison the graph for ridge regression has a concave shape (decreasing positive gradient). Thus we can assume it does not perform as well.

```
#misclassification rate for lasso


table(cut(coef(mod_l)[,1], breaks = 10))
```

```
##
##    (-3.63,-3.23]    (-3.23,-2.83]    (-2.83,-2.44]    (-2.44,-2.04]
##                1                0                0                0
##    (-2.04,-1.64]    (-1.64,-1.24]   (-1.24,-0.847]  (-0.847,-0.449]
##                0                0                0                0
## (-0.449,-0.0516]   (-0.0516,0.35]
##                0               39
```

```
preds_l_1se <- predict(mod_l, newx = test_X, type="response")
preds_l_min <- predict(mod_l, newx = test_X, type="response", s="lambda.min")

tab_l_1se <- prop.table(table(preds_l_1se > 0.5, test_Y))
tab_l_min <- prop.table(table(preds_l_min > 0.5, test_Y))
tab_l_1se
```

```
##        test_Y
##                 0         1
##   FALSE 0.4210234 0.1206897
##   TRUE  0.1240267 0.3342603
```

```
sum(diag(tab_l_1se))
```

```
## [1] 0.7552836
```

```
#miclassification rate for ridge
table(cut(coef(mod_r)[,1], breaks = 10))
```

```
##
##    (-4.5,-4.04]   (-4.04,-3.58]   (-3.58,-3.12]   (-3.12,-2.66]    (-2.66,-2.2]
##               1               0               0               0               0
##    (-2.2,-1.75]   (-1.75,-1.29]  (-1.29,-0.829] (-0.829,-0.371] (-0.371,0.0922]
##               0               0               0               0              39
```

```
preds_r_1se <- predict(mod_r, newx = test_X, type="response")
preds_r_min <- predict(mod_r, newx = test_X, type="response", s="lambda.min")

tab_r_1se <- prop.table(table(preds_r_1se > 0.5, test_Y))
tab_r_min <- prop.table(table(preds_r_min > 0.5, test_Y))
tab_r_1se
```

```
##        test_Y
##                 0         1
##   FALSE 0.4199110 0.1234705
##   TRUE  0.1251390 0.3314794
```

```
sum(diag(tab_r_1se))
```

```
## [1] 0.7513904
```

```
#miclassification rate for elastic net
table(cut(coef(e_mod)[,1], breaks = 10))
```

```
##
##   (-3.78,-3.38]   (-3.38,-2.98]   (-2.98,-2.59]   (-2.59,-2.19]   (-2.19,-1.79]
##               1               0               0               0               0
##    (-1.79,-1.4]   (-1.4,-0.999] (-0.999,-0.602] (-0.602,-0.205]  (-0.205,0.196]
##               0               0               0               0              39
```

```
preds_e_1se <- predict(e_mod, newx = test_X, type="response")
preds_e_min <- predict(e_mod, newx = test_X, type="response", s="lambda.min")

tab_e_1se <- prop.table(table(preds_e_1se > 0.5, test_Y))
tab_e_min <- prop.table(table(preds_e_min > 0.5, test_Y))
tab_e_1se
```

```
##        test_Y
##                 0         1
##   FALSE 0.4204672 0.1179088
##   TRUE  0.1245829 0.3370412
```

```
sum(diag(tab_e_1se))
```

```
## [1] 0.7575083
```

From the sum of diagonals we can see that the elastic net regression has a higher accuracy than the other two methods which have the same accuracy.

```
tab_r_min
```

```
##        test_Y
##                 0         1
##   FALSE 0.4215795 0.1140156
##   TRUE  0.1234705 0.3409344
```

```
tab_r_1se
```

```
##        test_Y
##                 0         1
##   FALSE 0.4199110 0.1234705
##   TRUE  0.1251390 0.3314794
```

```
r_coefs <- as.matrix(coef(mod_r, s="lambda.min"))
round(sort(r_coefs[r_coefs!= 0,] [-1]), digits = 3)
```

```
## MASQ02 MASQ35 MASQ03 MASQ20 MASQ17 MASQ25 MASQ12 MASQ40 MASQ34 MASQ08 MASQ28
## -0.106 -0.106 -0.093 -0.084 -0.048 -0.041 -0.035 -0.028 -0.010 -0.009  0.001
## MASQ19 MASQ06 MASQ26 MASQ09 MASQ15 MASQ23 MASQ36 MASQ33 MASQ39 MASQ29 MASQ32
##  0.015  0.017  0.022  0.029  0.036  0.036  0.044  0.048  0.050  0.052  0.053
## MASQ05 MASQ07 MASQ21 MASQ27 MASQ24 MASQ04 MASQ14 MASQ13 MASQ11 MASQ18 MASQ31
##  0.056  0.059  0.063  0.068  0.068  0.073  0.080  0.086  0.100  0.106  0.107
## MASQ38 MASQ22 MASQ37 MASQ30 MASQ01 MASQ16
##  0.113  0.125  0.136  0.150  0.158  0.281
```

```
l_coefs <- as.matrix(coef(mod_l, s="lambda.min"))
round(sort(l_coefs[l_coefs!= 0,] [-1]), digits = 3)
```

```
## MASQ02 MASQ03 MASQ35 MASQ20 MASQ17 MASQ25 MASQ12 MASQ33 MASQ29 MASQ32 MASQ39
## -0.116 -0.084 -0.075 -0.069 -0.053 -0.004 -0.001  0.005  0.016  0.028  0.029
## MASQ07 MASQ05 MASQ04 MASQ27 MASQ21 MASQ14 MASQ24 MASQ13 MASQ18 MASQ11 MASQ38
##  0.037  0.045  0.048  0.058  0.058  0.059  0.059  0.077  0.090  0.094  0.111
## MASQ31 MASQ22 MASQ37 MASQ01 MASQ30 MASQ16
##  0.124  0.145  0.149  0.189  0.190  0.420
```

```
e_coefs <- as.matrix(coef(e_mod, s="lambda.min"))
round(sort(e_coefs[e_coefs!= 0,] [-1]), digits = 3)
```

```
## MASQ02 MASQ03 MASQ35 MASQ20 MASQ17 MASQ25 MASQ12 MASQ23 MASQ09 MASQ36 MASQ33
## -0.106 -0.084 -0.082 -0.067 -0.043 -0.014 -0.009  0.004  0.006  0.009  0.024
## MASQ29 MASQ32 MASQ39 MASQ07 MASQ05 MASQ21 MASQ04 MASQ27 MASQ24 MASQ14 MASQ13
##  0.031  0.035  0.038  0.045  0.049  0.058  0.059  0.061  0.061  0.069  0.080
## MASQ11 MASQ18 MASQ38 MASQ31 MASQ22 MASQ37 MASQ30 MASQ01 MASQ16
##  0.094  0.097  0.111  0.118  0.137  0.145  0.172  0.178  0.363
```